Introduction to Programming and Data Structures

Python – Randomization

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

September, 2025

1 Randomization

Shuffling of data

```
import random
ls = [10, 20, 30, 40, 50]
random.shuffle(ls)
print(ls)
```

Output: [40, 20, 10, 50, 30]

Randomization of data

```
import random
random.randint(1, 100) # The interval is [1, 100]
```

Output: 15

```
import random
random.random() # The interval is [0.0, 1.0)
```

Output: 0.4289859005273219

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97

The interval is [1, 100)

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97 The interval is [1, 100)

```
import numpy as np
np.random.random(5) # From uniform distribution
```

Output: array([0.37076725, 0.10547203, 0.21298417, 0.96296838, 0.15390583]) The interval is [0.0, 1.0)

Random sampling with numpy

```
import numpy as np
np.random.randint(1, 100) # From uniform distribution
```

Output: 97 The interval is [1, 100)

```
import numpy as np
np.random.random(5) # From uniform distribution
```

Output: array([0.37076725, 0.10547203, 0.21298417, 0.96296838, 0.15390583]) The interval is [0.0, 1.0)

Note: Unlike the random() function in random module, the one in the random submodule of numpy module optionally takes size of the array.

Random sampling with numpy

```
import numpy as np
np.random.randn(5) # From normal distribution
```

Output: array([1.62029576, 0.29112406, 1.21198839, 0.26851418, -0.46712281])

```
import numpy as np
np.random.randn(3, 3) # From normal distribution
```

Output: array([[1.13217437, 0.56093212, -2.36792091],
[1.42652606, 0.68953983, 0.521175],
[1.36766496, 0.9488446 , -1.10042531]])

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Code 2:

```
import numpy as np
np.random.seed(20)
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Effect of the seed value on random sampling

Code 1:

```
import numpy as np
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Code 2:

```
import numpy as np
np.random.seed(20)
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
print(np.random.randint(1, 100))
```

Note: Unlike Code 1, Code 2 will generate the same set of random values whenever executed.

Drawing samples from different distributions using numpy

beta(a, b[, size])	– Samples from Beta distribution
binomial(n, p[, size])	– Samples from binomial distribution
chisquare(df[, size])	– Samples from chi-square distribution
dirichlet(alpha[, size])	– Samples from Dirichlet distribution
exponential([scale, size])	– Samples from exponential distribution
f(dfnum, dfden[, size])	– Samples from F distribution
gamma(shape[, scale, size])	– Samples from Gamma distribution
geometric(p[, size])	– Samples from geometric distribution
gumbel([loc, scale, size])	– Samples from Gumbel distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using `numpy`

hypergeometric(ngood, nbad, nsample[, size])	– Samples from hypergeometric distribution
laplace([loc, scale, size])	– Samples from Laplace distribution
logistic([loc, scale, size])	– Samples from logistic distribution
lognormal([mean, sigma, size])	– Samples from lognormal distribution
logseries(p[, size])	– Samples from logarithmic series distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using numpy

multinomial(n, pvals[, size])	– Samples from multinomial distribution
multivariate_normal(mean, cov[, size, ...)	– Samples from multivariate normal distribution
negative_binomial(n, p[, size])	– Samples from negative binomial distribution
noncentral_chisquare(df, nonc[, size])	– Samples from noncentral chi-square distribution
noncentral_f(dfnum, dfden, nonc[, size])	– Samples from noncentral F distribution
normal([loc, scale, size])	– Samples from normal distribution

Table: Functions in `random` submodule of `numpy`

Drawing samples from different distributions using `numpy`

pareto(a[, size])	– Samples from pareto distribution
poisson([lam, size])	– Samples from Poisson distribution
power(a[, size])	– Samples from power distribution
rayleigh([scale, size])	– Samples from Rayleigh distribution
standard_cauchy([size])	– Samples from standard Cauchy distribution
standard_exponential([size])	– Samples from standard exponetial distribution
standard_gamma(shape[, size])	– Samples from standard Gamma distribution
standard_normal([size])	– Samples from standard normal distribution

Table: Functions in `random` submodule of `numpy`

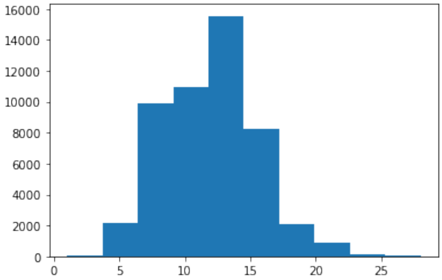
Drawing samples from different distributions using numpy

standard_t(df[, size])	– Samples from standard Student's t distribution
triangular(left, mode, right[, size])	– Samples from triangular distribution
uniform([low, high, size])	– Samples from uniform distribution
vonmises(mu, kappa[, size])	– Samples from von Mises distribution
wald(mean, scale[, size])	– Samples from Wald distribution
weibull(a[, size])	– Samples from Weibull distribution
zipf(a[, size])	– Samples from Zipf distribution

Table: Functions in `random` submodule of `numpy`

Visualizing a distribution

```
import numpy as np
sample = np.random.poisson(10, 50000)
import matplotlib.pyplot as plt
plt.hist(sample, 10)
plt.show()
```



Homework

1. Write a program in Python to simulate the rolling of a dice with 6 faces 1000 times. Count how many times each face (1–6) appears and display the results in a dictionary format.
2. Write a program in Python that generates a random password of length n (given by the user). The password should contain at least one uppercase letter from the English alphabet, one lowercase letter from the English alphabet, one digit (from 0 to 9), and one special character (from ˆ, !, @, , $, %, &, ∗).
3. Write a program in Python to simulate a random walk on the 2D grid as follows. Start at point (0, 0) and take 20 steps, such that each step makes a random move of one unit in one of the four directions (up, down, left, right). Print the final coordinates after 20 steps.